Software Engineering with Analysis Patterns

Michael Hahsler

Arbeitspapiere zum Tätigkeitsfeld
Informationsverarbeitung und Informationswirtschaft
Working Papers on
Information Processing and Information Management

Nr./No. 01/2001

Herausgeber / Editor:
Institut für Informationsverabeitung und Informationswirtschaft
Wirtschaftsuniversität Wien · Augasse 2-6 · 1090 Wien
Institute of Information Processing and Information Management
Vienna University of Economics and Business Administration
Augasse 2-6 · 1090 Vienna

Software Engineering with Analysis Patterns

Andreas Geyer-Schulz, Informationsdienste und Elektronische Märkte, Universität Karlsruhe (TH)
Michael Hahsler, Informationswirtschaft, Wirtschaftsuniversität Wien
hahsler@ai.wu-wien.ac.at

Abstract

The purpose of this article is twofold, first to promote the use of patterns in the analysis phase of the software life-cycle by proposing an outline template for analysis patterns that strongly supports the whole analysis process from the requirements analysis to the analysis model and further on to its transformation into a flexible design. Second we present, as an example, a family of analysis patterns that deal with a series of pressing problems in cooperative work, collaborative information filtering and sharing, and knowledge management. We present the step-by-step evolution of the analysis pattern *virtual library with active agents* starting with a *simple pinboard.*

In this paper we propose that using patterns in the analysis phase has the potential to reducing development time by introducing reuse already at the analysis stage and by improving the interface between analysis and design phase. To quantify our proposal we present results from the *Virtual University project of the Vienna University of Economics and Business Administration,* where the analysis patterns developed in this paper were used to implement several information systems. [1]

1 Introduction

The Internet still grows exponentially. A year has seven Internet years. The rapid change of technology in a networked world and the transition to the information society is a challenge, both for system developers and Internet users. This article makes two contributions for meeting the Internet challenge, namely it presents analysis patterns as a means to reduce time-to-market of systems for system developers, and the evolution of pinboard patterns to promote cooperation and collaboration between large groups of Internet users as an example for analysis patterns.

The paper is structured as follows: In section 2 we give a very brief overview over analysis patterns and propose a suitable template structure to describe analysis patterns. In sections 3 to 7 we present examples for analysis patterns for cooperative

[1] Andreas Geyer-Schulz and Michael Hahsler. Software engineering with analysis patterns. Technical Report 01/2001, Institut für Informationsverarbeitung und -wirtschaft, Wirschaftsuniversität Wien, Augasse 2-6, 1090 Wien, November 2001.

work and information sharing. In section 8 we analyse two information systems that are based on an implementation of the analysis patterns presented in this paper. For these examples we analyse code reuse to estimate the reduction of effort and development time using the well-known *Constructive Cost Model* by Boehm [Boehm, 1981].

2 Analysis Patterns

The term analysis pattern has been coined by Martin Fowler [Fowler, 1997] for patterns which capture conceptual models in an application domain in order to allow reuse across applications. Analysis Patterns, in contrast to Design Patterns, focus on organizational, social and economical aspects of a system, since these aspects are central for the requirements analysis and the acceptance and usability of the final system.

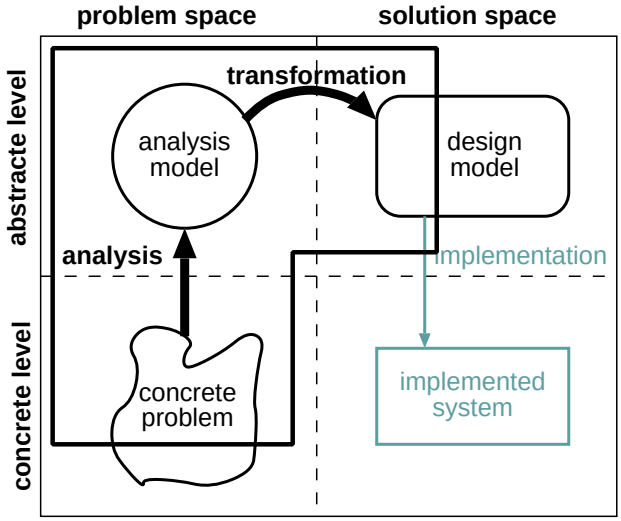

Figure 1: Analysis patterns in the software dvelopment process

Figure 1 shows the two main tasks where analysis patterns contribute to the software development process. First, analysis patterns speed up the development of abstract analysis models that capture the main requirements of the concrete problem by providing reusable analysis models with examples as well as a description of advantages and limitations. Second, analysis patterns facilitate the transformation of the analysis model into a design model by suggesting design patterns and reliable solutions for common problems.

In contrast to Fowler who prefers and uses a very free and informal format for pattern writing, we adopt a uniform and consistent format for describing analysis patterns which is easier to teach, learn, compare, write, and use, once the purpose of these sections has been understood. In table 1 we show the template which we propose for writing analysis patterns. It is used for the 4 example patterns in the following sections.

Pattern Name [Gamma *et al.*, 1995, Buschmann *et al.*, 1996] A pattern's name expresses the essence of a patterns precisely. It becomes part of the vocabulary used in analysis.

Intent [Gamma *et al.*, 1995] What does the analysis pattern do and what problem does it address?

Motivation [Gamma *et al.*, 1995] A scenario that illustrates the problem and how the analysis pattern contributes to the solution in the concrete scenario.

Forces and Context [Alexander, 1979] Discussion of forces and tensions which should be resolved by the analysis pattern.

Solution [Buschmann *et al.*, 1996] Description of solution and of the balance of forces achieved by the analysis pattern for the scenario in the motivation section. Includes all relevant structural and behavioral aspects of the analysis pattern.

Consequences [Gamma *et al.*, 1995, Buschmann *et al.*, 1996] How does the pattern achieve its objectives and what trade-offs exist?

Design [New] How can the analysis pattern be realized by design patterns? Sample design suggestions.

Known Uses [Gamma *et al.*, 1995, Buschmann *et al.*, 1996] Examples of the pattern found in real systems.

Table 1: Template for Analysis Patterns

Note, however, that our proposal preserves the typical context / problem / solution structure of patterns. With three exceptions (Forces, Solution, Design) the used sections originate from [Gamma *et al.*, 1995], albeit with a slight shift in meaning as required in the analysis phase and illustrated in table 1. The section on Forces is in the spirit of Alexander's patterns for architecture [Alexander, 1979], to which the usage of patterns in software engineering can be traced back. The section is used to discuss the forces and tensions which should be resolved by the pattern including social and economic conflicts.

The solution part of a pattern is from [Buschmann *et al.*, 1996]. It shows how to solve the problem and how a balance of forces is achieved by the pattern. It contains all diagrams which graphically describe the relevant structural and behavioral aspects of the pattern. We use the notation of the Unified Modeling Language (UML) [OMG, 1999].

The design part of an analysis pattern contains a description of a possible realization of the analysis pattern with one or several design patterns. The motivation for this section is to reduce software development time by providing design examples for the analysis pattern. In the best case, the design phase can be completely eliminated by reusing such a proposed design solution.

3 Examples: The Evolution from a Simple Pinboard to a Virtual Library with Active Agents

Next, we describe the rationale of the evolution from a simple pinboard to a virtual library with active agents. The driving forces of this evolution process are transaction cost reduction, scalability and performance improvements.

First, consider a simple pinboard. Simple pinboards constitute an efficient informal communication channel for small groups as long as the number of messages posted remains small. As soon as either the number of group members or the number of messages grows, group members need more and more time to keep track of the pinboard, searching specific information soon becomes an expensive, time-consuming task.

The evolution from a simple pinboard to a structured pinboard addresses these scalability problem by providing an environment for structured messages basically by extending the presentation interface and the message representation. These changes allow users to focus their attention only on small parts of the pinboard entries and improve the search capabilities considerably. However, there is a price to pay: properly using an information structure requires that either the user understands the semantic of the fields of the form he has to fill or that a professional librarian must classify the messages. In short, the cost, of the administration of such a pinboard rise as it grows and time goes by.

Moreover, frequently it is of advantage to have a pinboard for distributed information objects, e.g. because the owner of an information object prefers personal control

or because the owner changes the information object often or because a distributed object-store provides better load-balancing under heavy network traffic and a better performance with lower network-bandwidth. All of these factors reduce the cost of an organizations network and computing infra-structure. Clearly, the next two evolution steps are motivated by these issues.

The virtual library allows for distributed information objects with weak consistency checks. Adding agencies to the virtual library addresses the administration and maintenance costs inherent in structured pinboards and paves the way for new information services.

4 Analysis Pattern: A Simple Pinboard

Intent How can a dispersed group share information efficiently?

Motivation You are working for a large multinational company. This company has work-groups with its members dispersed over various countries. This groups have to work together to fulfill their jobs, therefore efficient communication and information sharing is a vital necessity for them.

It is not possible that all members of a work-group meet frequently, since this would be too time consuming. A work-group needs to exchange messages, communicate asynchronously and build up a common base of knowledge. Furthermore, to prevent information overload, the members of a work-group need an easy way to retrieve the needed information for a specific task. Unfortunately, existing communication media like letters, phone, fax and e-mail can not comply with all this needs.

A pinboard-like system (Figure 2) is used for the work-group. All members can read and compose messages. So every group member can use the messages when he needs them.

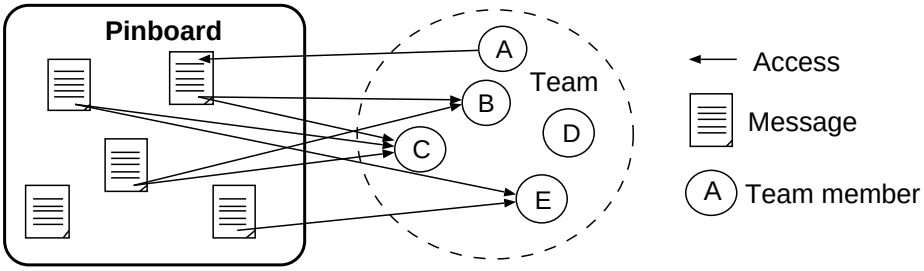

Figure 2: Simple Pinboard for Intra-Team Communication

Forces

- Efficient group communication is vital.

- Frequent meetings in person are not feasible.
- The risk of information overload is present.

Solution Use a pinboard for group communication. It stores messages from all group members and makes them available. For simple pinboards a message consists only of a textual body and the name of its author. To let members of the work-group choose the needed messages, full-text search over all messages has to be provided (e.g. search for messages posted from a specific group member or which contains a special keyword).

Figure 3 shows that the pinboard consists of three main components, the database for storing and retrieving messages, the interface providing access to users, and the control unit responsible for managing the whole pinboard.

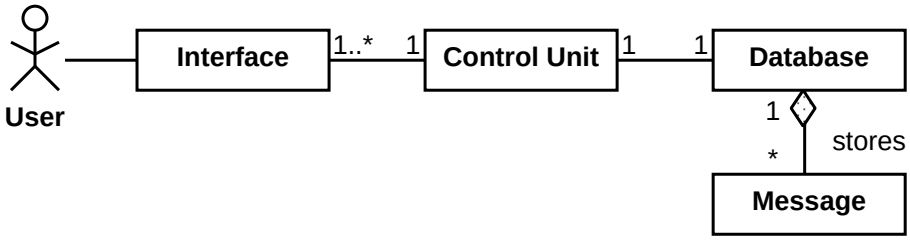

Figure 3: Structure of a simple Pinboard

The following actors use the system:

1. *User:* Retrieves messages from the pinboard.
2. *Information Provider:* Adds new messages to the pinboard.
3. *Administrator:* Is responsible for operating the Pinboard (e.g. deleting old messages from the database . . .).

Consequences The benefits and liabilities of the pinboard pattern are:

1. *Reduction of information overload.* The user actively searches for the information he needs. A problem occurs, however, if a messages has to be delivered quickly to a specific person. In this case alternative means of communication (e.g. electronic mail) should be used.
2. *Asynchronous communication.* Work is not permanently interrupted by incoming electronic mails or phone calls.
3. *Automatic generation of a collaborative information source.* Messages can be collected and archived to build a *collective dynabase* [Press, 1992].

4. *Confidential information.* If confidential information is shared via a pinboard, sufficient security precautions have to be taken.

5. *Lifetime of messages.* When is a message out of date and who deletes such a message? A pragmatic solution could be to automatically archive messages older than a defined age.

Design Although a pinboard seems to be easily implemented, several issues have to be considered in order to ensure reusability and extendability:

1. *Separation between the three components of the system.* For such a simple system like the Pinboard it is tempting to implement all parts in one module. This reduces implementation time since no interfaces between the components have to be specified, but it also makes later changes very expensive. For example, changing the interface from a proprietary client to a Web-based interface is very easy, if only the interface component has to be adapted.

2. *Extendability of the control unit.* To ensure that new functions can easily be added to the control unit, the best choice is to implement it using the *Interpreter* pattern [Gamma *et al.*, 1995, p.243] with an extendable instruction set.

3. *Choice of the database.* There are several possibilities to store messages, ranging from database management systems (object-oriented or relational) to simply using the file system. The decision should be based on search performance, license cost and implementation effort. However, if the database component is separated from the rest of the system, later change using the *Facade* pattern [Gamma *et al.*, 1995, p.185] is possible.

4. *Choice of the interface.* Here we have to choose between an interface for a proprietary client or a standard client like a Web-browser. The enormous advantage of Web technology is the availability of clients. It is common that they are distributed and preinstalled with every new operating system. The only reason for a proprietary client could be security considerations, but the built-in security features (like Netscape's Secure Socket Layer) are improving constantly and even a proprietary interface can be implemented e.g. using a Web-browser with a client side Java applet.

Known Uses

- News Network Transfer Protocol [Kantor and Lapsley, 1986].
- Bulletin Boards (e.g for group ware applications like the BSCW system [Bentley *et al.*, 1997]
- Guest books of Web-sites.

5 Analysis Pattern: Structured Pinboard

Intent Collect, group and present structured information.

Motivation For your work-group collecting simple text messages is not sufficient to find the needed information within reasonable time. This is due to the size of the group and thus the number of messages available as well as to the variety of different topics the group has to deal with.

It is important, that a user can decide quickly if the message contains what he is searching for. Also grouping by specific criteria (e.g. by several categories) makes finding related information easier. The simple Pinboard only supports unstructured text, but different people use different terms to describe the same concept. Therefore all problems of information retrieval are present.

We use a pinboard that supports structured messages to address these problems. Additionally to the textual body you can use separate searchable fields for the subject, keywords and several predefined categories (e.g. the information's language, or a special classification scheme). To compose messages, predefined forms are used.

Forces

- Big groups with a great number of messages and/or topics make a simple pinboard confusing.
- It takes more time to read unstructured text.
- Unstructured text is hard to read and very hard to process automatically.
- People tend to forget important aspects in their messages.

Solution Build a pinboard with structured messages. With such a system that provides several semi-structured message types, *intelligent information sharing* is possible [Malone *et al.*, 1987].

Figure 4 shows, that structured messages are obtained by composing the `Message`-objects of several `Field`-objects. To provide a more flexible interface, the `Interface` delegates presentation issues to several subclasses using the Strategy or Template pattern [Gamma *et al.*, 1995]. While Interface A in Figure 4 could be a simple search interface, Interface B could provide access via e.g. a hierarchical classification tree, like NAICS [NTIS, 1997].

Consequences The benefits and liabilities of the Structured Pinboard pattern are:

1. *Structure.* Understanding information with a logical structure is easier and takes less time than reading unstructured text.

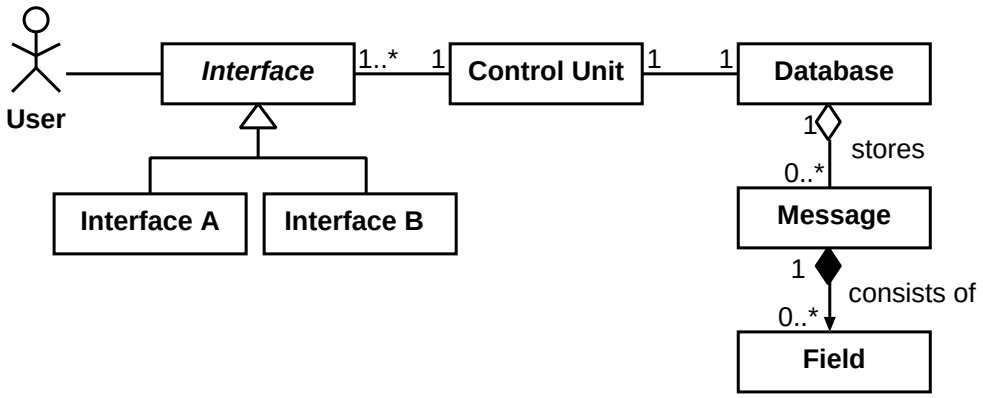

Figure 4: Structure of a Structured Pinboard

2. *Automatic processing.* It is possible to use fields whose content is restricted to a predefined set of values. With such fields automatic processing can be employed without complex information recognition techniques.

3. *Completeness of information.* Presenting the user a form with all mandatory fields marked, prevents him from omitting essential information.

4. *Incompatibility between different types of messages.* Using a fixed form to compose messages has the drawback that all possible fields have to be predefined. No new fields, that would improve the readability of the message, can be added by the user while writing it. And if several different forms with different logical structures are provided, one has to find a strategy how to deal with polymorphic message types.

Design Additional to the issues presented for the simple Pinboard the following points are important:

- *Development of the message form.* The implementation of the form requires a detailed analysis of the information needed. The form has to be usable intuitively (without looking into the manual), otherwise the users will loose time thinking about which information should go into which field. If some fields could be easily misunderstood, their purpose has to be explained in the form.

- *Extension of the Interface.* With structured messages it is possible to improve the interface considerably. E.g. the message can be presented by the subject and the author only, therefore more messages can be displayed on one screen. Another improvement is possible by providing an interface that lets you search only in a specific field or lets you restrict the search to messages that contain a specific value in one field (e.g. the field containing the date of a message could be restricted to dates not older then one week while searching for a keyword).

Known Uses

- The simple Pinboard pattern is a special case of the Structured Pinboard with only one field, the textual body. Therefore all applications for simple Pinboards can also be realized using a Structured Pinboard.
- The Information Lens system [Malone *et al.*, 1987].
- Broker services on the Internet (e.g. on-line job offers of an employment agency).

6 Analysis Pattern: Virtual Library

Intent A Virtual Library is used to provide easy access to distributed information sources.

Motivation At a university much research and teaching material is available on-line, but most of it is hard to find since it is dispersed all over the campus network. Typically some material is accessible from departments' home pages or from the lecturers' personal home pages. Unfortunately, often material is only available from pages deep in the Web-space or it is not linked at all. Even though most universities have search engines, presenting all material in an organized way would improve accessibility and thus increase its usage and value.

A common approach to cope with the problem of dispersed information is to centralize it, for example, in a digital library. A digital library is a depository for storing and retrieving documents. However, there are several reasons why the introduction of a digital library can be problematic or even fail.

- Documents that tend to change frequently (topical lecture material) produce additional administrative overhead of storing the document every time it is changed.
- Multimedia supported documents are hard to store in a digital library while preserving their functionality (e.g. lecture material that uses server side programs to visualize a model manipulated by the user).
- In organizations with many virtually independent units it is hard to convince all that a central storage of information is favorable for every member of the organization and not just a means of centralizing power.

  We use a central inventory as shown in Figure 5 that provides uniform access to material and information resources available on the campus network (or the Internet). The inventory consists of index cards which additionally to meta information (e.g. the information object's name, keywords, a description ...)

[Weibel *et al.*, 1998] also contain a reference (e.g. a hyperlink) to the real information object they represent.

Use separate fields of the Structured Pinboard's message class to store the references and its associated meta information. An important issue is that references are only valid as long as the referenced information objects are available. So it is necessary to implement a mechanism to ensure that all references are checked frequently for consistency.

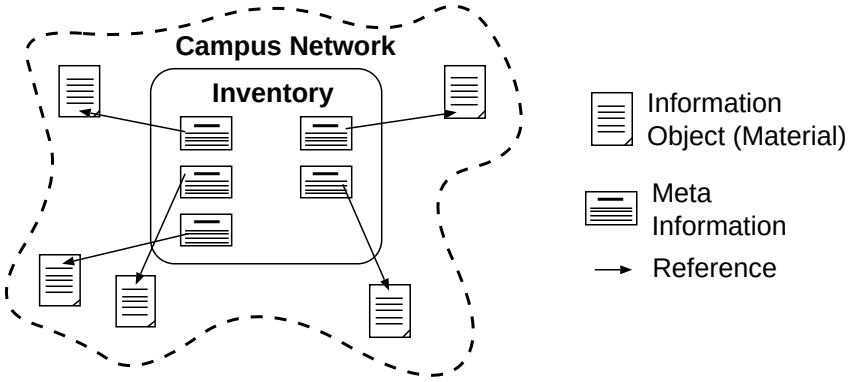

Figure 5: A Virtual Library

Forces

- Need for a central access point to dispersed information.
- Independent information provider with different objectives and using different technologies.
- High change rate of the provided information.

Solution The virtual library uses the structure of the structured pinboard. The pinboard's messages are used to store meta information about the information objects, as well as a reference to them.

Consequences In addition to the benefits and liabilities known from the structured pinboard pattern, the virtual library pattern has the following consequences:

1. *Information objects physically stay with their owners.* Since only meta information with a reference is stored centrally, the owner of the real object can manipulate it without any restriction. This is of enormous advantage, if frequent updates of the objects are necessary.

2. *Consistency problems.* Information objects can be deleted or moved to other locations without notifying the central directory. Therefore all references and the meta information have to be checked frequently in order to preserve system consistency. Moreover, repairing invalid references can be a very expensive task.

Design The Virtual Library uses a Structured Pinboard which has an extendable control unit. Therefore, adding new functions (e.g. consistency check, other administrative function) is possible without changes. A consistency check of references to information objects as well as additional functions (e.g. automatic keyword extraction) can be implemented, as shown in Figure 6, using the *Visitor pattern* [Gamma *et al.*, 1995].

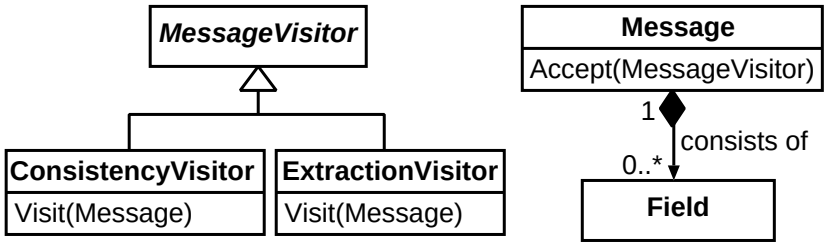

Figure 6: Structure of additional functions for a virtual library

Known Uses

- Directory based services (e.g. the WWW Virtual Library, Yahoo!).
- The Virtual Library system [Hahsler, 1997] developed for the Living Lectures-Virtual University Project (`http://vu.wu-wien.ac.at`).

7 Analysis Pattern: Virtual Library with Active Agents

Intent Active agents of a virtual library are used to reduce the transaction cost of library users in collecting, classifying, maintaining, searching, using and reusing information sources.

Motivation You are working for a research-group on e-commerce and you are responsible for keeping track of new product developments and product launches of competitors in the global market place.

The exponential growth and fast rate of change on the Internet makes technology monitoring and maintenance of knowledge-bases increasingly expensive and infeasible. For research groups in industry and universities there is a growing need for timely

and accurate high-quality information cheaply available. Unfortunately, current search engine technology is not yet able to deal with this problem.

Active agents which automate the day-to-day maintenance and observation tasks as well as the adaptation of the user-interface to a changed info-base can be employed to cope with growth and to improve information quality.

Forces

- Growth in the number or of the size of information objects.
- Need of timely, high-quality information.

Solution Extend a virtual library with active agents to perform all necessary tasks. These active agents use the agent analysis pattern of Russell and Norvig [Russell and Norvig, 1995]. Figure 7 shows how two of several possible active agents collaborate for a virtual library. The environment, which consists of the virtual library with its meta-information, the referenced information objects and the users, is perceived by the agents via their sensors. The agents gather information and influence their environment by updating information of the virtual library (observation agents) or passing results to users (interface agents). In Figure 7 the whole task of observing a distant information object and presenting the results of the observation to a user is divided between an observation agent and an interface agent which act independently from each other. In Figure 7 this is denoted by the message sequences A1-A3 and B1-B3. Note, that this results in a weakening of consistency constraints which usually improves performance, reduces resource requirements (e.g. network bandwidth), and simplifies the implementation of the system.

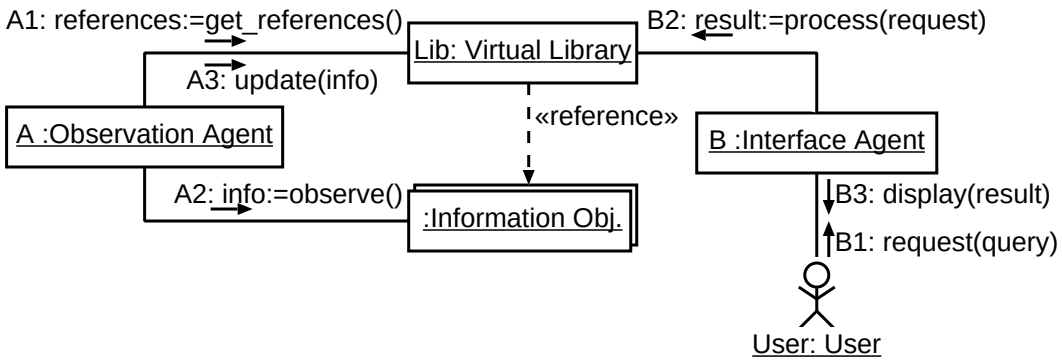

Figure 7: Collaboration in an Agency for Virtual Libraries with Active Agents

Consequences The benefits and liabilities of the virtual library with active agents pattern are:

- Reduction of transaction cost, because of the agent's automated information services.
- A challenge to find an acceptable compromise between synchronization of agents and consistency of information.

Design The virtual library with active agents uses a virtual library with a scheduler for agent processes. Thought must be given to the trade-off between synchronizing agents and improved consistency. For agents e.g. the *Observer* pattern [Gamma *et al.*, 1995, p.293] can be used.

Known Uses

- Consistency check and labeling [Geyer-Schulz *et al.*, 1999], [Geyer-Schulz and Hahsler, 2000].
- News Wire Services [Oki *et al.*, 1992].
- Recommender Systems [Resnick and Varian, 1997]

8 Benefits of Analysis Patterns

We used the analysis patterns described in this paper to realize several information systems for the *Virtual University project at the Vienna University of Economics and Business Administration* (see: `http://vu.wu-wien.ac.at`). First, we implemented the analysis pattern *virtual library with active agents* in the programming languages PERL and HTML [Hahsler, 1997], and then we reused the resulting software to build several information system for different tasks within the virtual university project.

To quantify the benefit of the used analysis patterns we compare the actual effort needed to implement two information systems for the project reusing the implementation of the patterns with an estimate of the effort necessary to build these systems from scratch. For the estimate we use the well-known *Constructive Cost Model (COCOMO)* [Boehm, 1981, Boehm *et al.*, 2000] to convert *Lines of Code (LOC)* into effort measured in man months (MM). We use the parameters for the basic model of COCOMO suggested by Boehm in the *Organic Mode of Software Development* since the analyzed information systems were developed by small teams and the technical specifications were not fully fixed at the beginning of the development. This provides us with the simple formular shown in equation 1, where 2.4 and 1.05 are the parameters of the model, $KDSI$ is the number of delivered source instructions (a similar measure like LOC) in 1000, and MM is the estimated amount of man months needed to develop the system.

$$MM = 2.4 \times KDSI^{1.05} \tag{1}$$

The analyzed information systems of the project, their size, the size of the development teams and the actual effort needed to implement them are shown in figure 2. For the calendar of events we changed the interface of the virtual library, the structure of its meta information and we added an automatic archiving function for events that already took place. For the digital library we had to extend the virtual library with a repository for documents, implement support for polymorphic meta information and we had to change and add many administrative functions (e.g. upload of documents, management of documents in several formats, full text search in the documents).

Project	**Size** in LOC	**Team Size**	**Actual Effort** in MM
Calendar of Events	4021	1	0,5
Digital Library	7616	2	6

Table 2: The analyzed projects

The calendar of events. The on-line calendar of events is a software for the collaborative collection of web sites for events, which perfectly is covered by the analysis pattern *virtual library.* Therefore, we could reuse most of the implementation of the pattern. In table 3 the percentage of the code reuse is summarized. More than 93% of the LOC in PERL and 69% of the LOC in HTML (with embedded PERL statements) could be reused unchanged. By using COCOMO to estimate man months from LOC, this reuse resulted in a reduction of the total development effort of more than 9 man months or over 90%.

	Unit	**Total Code**	**Reused Code**	**Code Reuse**
PERL	LOC	3743	3502	93,56%
HTML	LOC	278	192	69,06%
Sum	LOC	4021	3694	91,67%
Sum	MM	10,35	9,46	91,40%

Table 3: Code reuse for the calendar of events

The development of a digital library. For the development of the digital library based on the implementation of the analysis pattern *virtual library,* several changes were necessary. Virtual libraries only manage meta information and references to documents. Therefore, the design of the virtual library had to be augmented significantly by components necessary to store and manage the actual documents. Figure 8 shows the main changes.

But not only the design changed, also the analysis presented in the pattern *virtual library* only applies partial to a digital library. Virtual libraries are used to provide

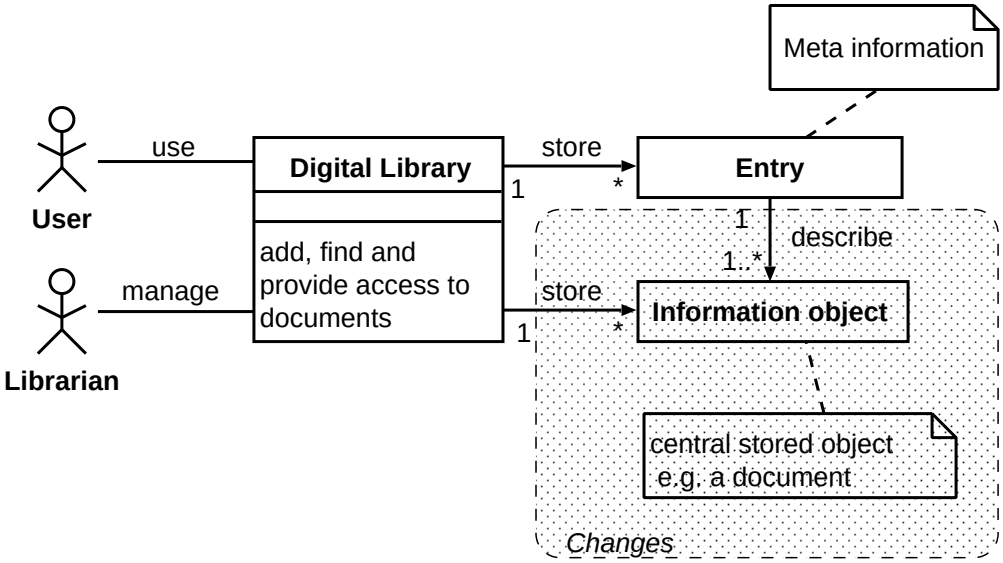

Figure 8: Changes from a virtual library to a digital library

access to distributed information sources, in contrast, digital libraries provide a central storage for information in form of documents. This leads to a different set of forces and consequences for a digital library. In fact, digital library should become its own analysis pattern.

However, analysis patterns, as all patterns, are supposed to lead to clearer, and more flexible and reusable design and code. Thus, the implementation of the virtual library should be easy to change and extend to build a digital library. Table 4 summarizes the results of the code reuse for building a digital library from the virtual library. More than 53% of the LOC in PERL and more than 33% of the LOC in HTML could be reused unchanged. This resulted in a total code reuse of over 51% or an estimated reduction of effort of 10 man month, which is about 50% of the total development effort.

	Unit	**Total Code**	**Reused Code**	**Code Reuse**
PERL	LOC	6954	3687	53,02%
HTML	LOC	662	224	33,84%
Sum	LOC	7616	3911	51,35%
Sum	MM	20,23	10,05	49,68%

Table 4: Code reuse for the development of a digital library

Only a small part of the potential benefits of analysis patterns could be shown by the simple analysis of code reuse presented in this section. Analysis patterns provide a

tool to save time and effort in the analysis phase, which only can be correctly measured by comparing the costs due to analysis phase and its effects with and without using patterns. However, this first evaluation of code reuse provides strong evidence, that patterns already used in the analysis phase, accompanied by design patterns have the potential to significantly reduce time-to-market as well as costs in the long run.

9 Conclusion

The two main contributions of this article are:

- Four analysis patterns and the economic rationale for the evolution, which address problems in cooperative work, collaborative information filtering and sharing, and knowledge management.
- An improved template for analysis patterns with a clear interface to the design phase to facilitate reuse.

To provide evidence for these contributions we analyzed the code reuse for two applications of the analysis pattern *virtual library*, presented in this paper, which revealed a significant reuse potential (between 49 and 91%).

Further and more detailed research on the benefits of patterns in all phases of the software life-cycle is still necessary.

References

[Alexander, 1979] Christopher Alexander. *The Timeless Way of Building*, volume 1 of *Center for Environmental Structure Series*. Oxford University Press, New York, 1979.

[Bentley *et al.*, 1997] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, S. Sikkel, J. Trevor, and G. Woetzel. Basic Support for Cooperative Work on the World Wide Web. *International Journal of Human-Computer Studies*, 46(6):827–846, June 1997. "Special issue on Innovative Applications of the World Wide Web".

[Boehm *et al.*, 2000] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II*. Prentice Hall PTR, Upper Saddle River, NJ, 2000.

[Boehm, 1981] Barry W. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Buschmann *et al.*, 1996] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern–Oriented Software Architecture, A System of Patterns*. John Wiley & Sons Ltd, Chichester, 1996.

[Fowler, 1997] Martin Fowler. *Analysis Patterns: Reusable Object Models*. Object Technology Series. Addison–Wesley Publishing Company, Reading, 1997.

[Gamma *et al.*, 1995] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object–Oriented Software*. Addison–Wesley Professional Computing Series. Addison–Wesley Publishing Company, New York, 1995.

[Geyer-Schulz and Hahsler, 2000] Andreas Geyer-Schulz and Michael Hahsler. Automatic labelling of references for information systems. *in: W. Gaul et al. (Eds.), Procs. of the 23rd Annual Conference of the GfKl, Bielefeld*, 2000. To appear.

[Geyer-Schulz *et al.*, 1999] Andreas Geyer-Schulz, Michael Hahsler, and Georg Schneider. The virtual university and its embedded agents. *ÖGAI Journal*, 18(1):14–19, 1999.

[Hahsler, 1997] Michael Hahsler. Software Patterns: Pinwände. Masters Thesis, Vienna University of Business Administration and Economics, November 1997.

[Kantor and Lapsley, 1986] Brian Kantor and Phil Lapsley. Network News Transfer Protocol. Request for Comments 977, Network Working Group, February 1986.

[Malone *et al.*, 1987] Thomas W. Malone, Kenneth R. Grant, Franklyn A. Turbak, Stephen A. Brobst, and Michael D. Cohen. Intelligent Information-Sharing Systems. *Communications of the ACM*, 30(5):390–402, May 1987.

[NTIS, 1997] NTIS. *North American Industry Classification System (NAICS) - United States*. National Technical Information Service, Springfield, 1997. ISBN-0-934213-57-7.

[Oki *et al.*, 1992] B. M. Oki, D. Goldberg, D. Nichols, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[OMG, 1999] OMG. Unified modeling language. Specification v1.3, Object Management Group, June 1999. http://www.omg.org/technology/uml/.

[Press, 1992] Larry Press. Collective Dynabases. *Communications of the ACM*, 35(6):26–32, June 1992.

[Resnick and Varian, 1997] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, March 1997.

[Russell and Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, 1995.

[Weibel *et al.*, 1998] Stuart Weibel, John Kunze, Carl Lagoze, and Misha Wolf. Dublin Core Metadata for Resource Discovery. Request for Comments 2413, Network Working Group, September 1998.